CLAIMS

1.   A method for loop optimization within a dynamic compiler system, comprising:

executing a computer program having a loop structure, wherein the loop structure
5   includes a loop exit test to be performed during each loop iteration;

compiling the loop structure during the execution of the computer program; and

creating an unrolled loop structure during the compiling operation, wherein the
unrolled loop structure includes plurality of loop bodies based on the original loop
structure.

10

2.   A method as recited in claim 1, wherein the unrolled loop structure
includes the loop exit test.

3.   A method as recited in claim 2, wherein the loop exit test is performed
15   once for each iteration of the plurality of loop bodies.

4.   A method as recited in claim 1, further comprising the operation of
building a loop tree based on loops included in the computer program.

5.   A method as recited in claim 4, wherein nested loops are represented in the loop tree as child nodes.

6.   A method as recited in claim 5, wherein parallel loops are represented in the loop tree as nodes on a same level of the loop tree.

7.   A method as recited in claim 1, further including the operation of performing loop clean up.

8.   A method as recited in claim 7, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

9.   A method as recited in claim 7, wherein the loop clean-up includes optimizing nested loop structures having invariant operations.

10.   A dynamic compiling system, comprising:

an interpreter capable of interpreting instructions of a computer program during execution of the computer program, the interpreter being further capable of requesting that a particular instruction be compiled; and

a compiler capable of compiling the instructions as requested by the interpreter, wherein the compiler is further capable of creating an unrolled loop structure when compiling an original loop structure of the computer program, wherein the unrolled loop structure includes plurality of loop bodies based on the original loop structure.

5

11.   A dynamic compiling system as recited in claim 10, wherein the unrolled loop structure includes the loop exit test.

12.   A dynamic compiling system as recited in claim 11, wherein the loop exit test is performed once for each iteration of the plurality of loop bodies.

13.   A dynamic compiling system as recited in claim 10, wherein the compiler is further capable of building a loop tree based on loops included in the computer program.

15

14.   A dynamic compiling system as recited in claim 13, wherein nested loops are represented in the loop tree as child nodes.

15.   A dynamic compiling system as recited in claim 14, wherein parallel loops are represented in the loop tree as nodes on a same level of the loop tree.

20

16.   A computer program embodied on a computer readable medium for loop optimization within a dynamic compiling, comprising:
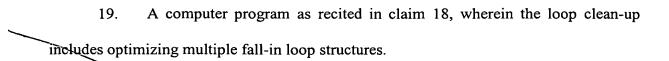
a code segment that interprets a computer program having a loop structure,

5   wherein the loop structure includes a loop exit test to be performed during each loop iteration;

a code segment that compiles the loop structure during the execution of the computer program; and

a code segment that creates an unrolled loop structure during the compiling

10   operation, wherein the unrolled loop structure includes plurality of loop bodies based on the original loop structure.

17.   A computer program as recited in claim 16, wherein the unrolled loop structure includes the loop exit test, and wherein the loop exit test is performed once for

15   each iteration of the plurality of loop bodies.

18.   A computer program as recited in claim 17, further comprising a code segment that performs loop clean up.

19.   A computer program as recited in claim 18, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

20.   A computer program as recited in claim 19, wherein the loop clean-up

5   includes optimizing nested loop structures having invariant operations.